A short primer on cryptography

A. V. Atanasov

April 14 2007

1 Preliminaries

(This section is an introduction to the referred mathematical concepts. Feel free to skip it if you are familiar with the first principles of number theory)

When young children are first introduced to mathematics, one of the main topics of interest is integers and operations among them. It is later that the teacher asks the question, "Are there numbers between numbers?" and its answer gives rise to the whole new universe of real numbers. For the purposes of this discussion, we will confine ourselves to the integers. When a student is first introduced to number theory, she might initially think that "whole numbers" are easy and there are no curious problems concerning them — there is nothing further from the truth. Elementary number theory is a fascinating branch of mathematics, and certainly the most romantic one. K. F. Gauss refers to arithmetic as "the queen of mathematics".

Before going into the subject, let us briefly discuss a few issues of notation. It is universally accepted that modern mathematics is built on top of set theory. One can think of a set as a collection of objects (e.g. fruit, numbers, pastry shops or functions). We will denote as $\mathbb{N}$and $\mathbb{P}$the sets of positive integers (also called natural numbers) [1] and prime numbers respectively. The sign $\in$ means *element of* or simply *is* (e.g. $3 \in \mathbb{N}$ means 3 is a natural number), whereas $\notin$ denotes the opposite (e.g. $2.5 \notin \mathbb{N}$ means that 2.5 is not a natural number). [2] Without further ado, let us chart the basics of number theory.

1.1 Remainders and Congruences

Recall a third grade class, when you first experienced division. We were not told that $5/2 = 2.5$ but that it equals 2 with a remainder 1. In order to properly think in the context of number theory [3] , please forget about the existence of real numbers, think integers, imagine you are in the third grade. When a number, a, divides another, b, with remainder 0 we say that a divides b and write it $a|b$ for short (e.g. $3|6$). When two numbers, a and b, yield the same remainder when divided by some fixed number n, we say that a and b are congruent modulo n and abbreviate it $a \equiv b \pmod{n}$ (e.g. $6 \equiv 21 \pmod{5}$). Finally, we call a number prime if it is greater than one and

[1] From now on "number" will generally refer to a positive integer unless otherwise stated.

[2] Almost every book on elementary number theory and abstract algebra devotes its first chapter to the preliminary notions of naïve set theory. For such an account please refer to [2] . For an axiomatic account see [4] . It might also be beneficial to look at [3] (a literary pearl and the winner of Pulitzer Prize) which includes numerous discussions (between Achilles and the Tortoise) on the foundations of mathematics.

[3] Please refer to [5] for a more complete introductory account of elementary number theory.

its only divisors (positive) are 1 and itself (e.g. 1, 5 are the only divisors of 5, so $5 \in \mathbb{P}$). Otherwise, we call a number composite (e.g. all 1, 2, 5 and 10 are divisors of 10, so $10 \notin \mathbb{P}$). Suppose there is a universe whose people are "number-blind" with respect to some fixed number n, so that they cannot tell the difference between two numbers if they are congruent mod n. Then $0, n, 2n, \ldots$ would all be considered the same number, simply called 0, and 1 would be $1, n+1, 2n+1, \ldots$, etc. This universe where only remainders mod n are considered, even when adding, subtracting, multiplying and dividing, is called the *finite field of n elements*, denoted $\mathbb{F}_n$ (e.g. $\mathbb{F}_{10}$, $5+7=2$). Computers inherently work with 0s and 1s, and this is why elements of $\mathbb{F}_2$ (0 and 1) are called *bits*.

1.2 Complexity of Algorithms

Informally, an algorithm means a structured computational method which solves a certain problem. Notice that for a given problem there might be more than one algorithm which solves it, possibly with different characteristics. In this section we will take a glimpse of the underlying methods which inevitably appear throughout cryptography. There will be no formal attempt at the subject. When it comes to execution of the algorithm, which is carried out by computers, we are interested in two factors, namely how long it will take, and how much space (memory) it require. These two entities, which are referred to as time and space complexity, are often expressed as functions of the input (or its size). In order to avoid a lot of dry theory, we would take a practical example. Let A, B and C be three algorithms taking time N, N^2 and 2^N seconds respectively where N is the input (these are formally referred to as linear, quadratic and exponential complexities). Suppose we run all three algorithms for $N = 10$ and twice as much. The time A takes will double, that of B will quadruple and the time of execution of C will be multiplied by a factor of more than 1000. If someone had started C at the big-bang with an input of merely $N = 62$, it would still not have finished executing! Theoretically it is not impossible to compute C, but it is practically infeasible. The security of cryptography is ensured by functions (problems) which are easy to compute (like A and B), but their inverses are infeasible to find (like C). For example, it is relatively easy to multiply two prime numbers even if they are large, but factorizing an integer whose prime multiples are large is quite difficult. The mentioned example is known as the *factorization problem*. All solutions known are infeasible to compute time-wise, but we have no proof that there is no fast algorithm. Another such pair of easy-hard problems is powering and taking logarithms in $\mathbb{F}_p$ for $p \in \mathbb{P}$. Raising a number to some power can be done in time which depends on the logarithm of the power and the square of the length of the number ($(\log p)^3$ in general), but the reverse operation, called *discrete logarithm*, is very slow for large numbers (around 200 digit prime numbers p).

2 Introduction to Cryptography

Cryptography grew as an attempt to secure information and communications. In *The Histories*, Herodotus describes the first account of people using secret writing — in the fifth century B.C. the Greek employed obfuscated communication techniques to receive information from Persia about Xerxes' plan to invade them. Despite the long history of demand for such methods, little progress occurred before the twentieth century. Only after people realized how intimately related cryptography is to mathematics and number theory in particular did real advance in the direction of true security happened.

2.1 Private Key Systems

If the average person is asked to describe how she imagines a cryptographic system, the most common answer involves a two phase algorithm, one that encrypts the input text (called plain-text) on the basis of some key, text or number [4] , and a reverse operation that decrypts correctly the encrypted plain-text (called cipher-text) provided the same identical key. If the key is not available, it should be infeasible to reverse the operation. This mechanism is called *Private Key Cryptography* or *Symmetric-Key Cryptography*. Let us imagine that Alice and Bob are in possession of a common key, which only they know. Assuming that Eve might wire-tap their phone calls, when Alice calls Bob she always encrypts her messages with the key. On the other side Bob decrypts the received messages using his copy of the key obtaining back the human-readable form. Eve will not be able to understand their conversation, because she would not know the key, and therefore would not be able to decrypt. When Bob speaks to Alice, the same conventions are used. It is understandable why this communication protocol is characterized as symmetric.

2.2 Public Key Systems

There is one very obvious flaw in the above mechanism. How do Alice and Bob agree on the key and exchange it? If they meet physically then it is very easy to do that. But if they do not, this is not such an easy matter. If Alice randomly generates a key and sends it to Bob, Eve might be tapping the communication line and then she would be in possession of the key too, which in turn means that their communication is not secure anymore.

This problem of key exchange is clearly a major threat to the security of communication. It is not true that two parties are not able to securely exchange keys. However, nobody knew how to perform this operation until recently. The first discovery of such a procedure was in the early 70s at Government Communications Headquarters – GCHQ (the British equivalent of NSA). Of course, the information was not released in the public domain. Whitfield Diffie and Martin Hellman reinvented the procedure in 1976 which is nowadays known as Diffie-Hellmann key exchange [5] . The invention of numerous such schemes followed some of which even do not require a common key but an entangled use of two pairs of keys — one public and one private for each party (the schemes widely used nowadays are of this type). Examples of such are RSA and ElGamal, which fall under the general category of *Public Key Cryptography* systems. The description of these goes beyond the scope of this text.

3 Diffie-Hellmann Key Exchange

In this section we describe the Diffie-Hellman key exchange. To overcome gradually the various technical difficulties, we start by presenting the idea symbolically, which is followed by a short discussion of its realization.

[4] Plain-text, cipher-text and keys are often represented in text form (although unreadable) for human and length reasons. On the other hand, mathematics manipulates numbers. Trivial conversion mechanisms are used between textual and numerical representations. These will not be discussed here, and we will predominantly work with numbers for computational and demonstration purposes.

[5] See [1] for the original paper.

3.1 Conceptual Ideas

Although the name refers to the scheme as a "key exchange", there is no real exchange of a single functional key. Conceptually, in the end both parties compute the same key and nobody else is able to do so. Suppose both Alice and Bob generate their own keys, respectively A and B, which they strictly keep to themselves. Also they need to know a transformation function of a key, which is easy to compute but difficult to invert, denoted T(key). Alice and Bob exchange their transformed keys in a way that they respectively learn $T(B)$ and $T(A)$. Then every party is knowledgeable of its own key and the transformed version of the other party's key. Suppose also we also know of a "mixing" operation performed on a key and the transformed version of another, denoted M(key, transf. key). Furthermore, this mixing should be symmetric, in the sense that for every two keys X and Y, we require $M(X, T(Y)) = M(Y, T(X))$. In the above setting, Alice can mix her key with Bob's by computing $K = M(A, T(B))$, and Bob will be able to compute his own copy of the same key by $K = M(B, T(A))$, which concludes the exchange. We observe that Eve could not compute K, because the only information she has is $T(A)$ and $T(B)$, and the mixing procedure requires at least one authentic key which cannot be obtained due to the complexity of inverting T.

3.2 Realization

The public information information necessary for a Diffie-Hellmann system is a prime number, p (the larger the more secure), and a number g such that $1 < g < p$ for which some additional constraints hold [6] . The private keys of Alice and Bob are respectively the numbers a and b such that $1 < a < p - 1$ and $1 < b < p - 1$ (to avoid security breaches a, b should not be chosen very close to 1 or $p - 1$ i.e. somewhere in the middle 90%). Then Alice sends to Bob g^a, and respectively receives from him g^b (exchange of transformed versions). Once Alice has g^b, she raises it to the power of her own key to obtain the key $k = (g^b)^a = g^{ab}$ (mixing). Similarly Bob computes $k = (g^a)^b = g^{ab}$. Now both parties know the common key $k = g^{ab}$. Yet, Eve has only learned g^a and g^b which is not enough to compute g^{ab}, unless she can compute the discrete logarithm of one of the values she has to obtain a or b respectively. It is clear that the security of the scheme depends on the discrete logarithm problem. Modern standards of security require use of a prime number p of size about 200 decimal digits. Finding such a p and a primitive root for it, g, is not an easy task which is beyond our discussion.

4 Ideas of Security

The security of modern cryptography mainly depends on the difficulty of performing integer factorization, computing discrete logarithms and solving other similar problems. Choosing long enough keys is one of methods that ensures no third party is aware of the transmitted message. However, computers become faster by the hour, and there is continuous progress on code-breaking techniques (called *cryptanalysis*) which makes most of the keys used 20 years ago utterly insecure. Also, although we described operations in $\mathbb{F}_p$ as fast and easy (relatively to the difficult problems),

[6] Technically, g should be a primitive root in order to avoid security breaches. A primitive root is a generator for $(\mathbb{Z}/p\mathbb{Z})^\times$, which is the multiplicative structure behind $\mathbb{F}_p$. In simpler words, $g^s \neq 1$ for any $1 \leq s < p - 1$. The least such $s \geq 1$ is called the order of g, therefore primitive roots have order $p - 1$ which is the size of $(\mathbb{Z}/p\mathbb{Z})^\times$. For more information on the topic, please consult any introductory text in abstract algebra, such as [2] .

these are quite time consuming considering the large amounts of data (e.g. a secure ElGamal to today's standards requires to raise a 200 digit number to the power of another 200 digit number roughly for every sentence transmitted). This problems is often addressed using secure communication tunnels (channels) (the secure hypertext protocol, *https*, uses a secure channel known as *SSL*, over the insecure Internet network). We already covered the basics of tunneling, which is key exchange. In practice, the parties use a public key encryption system to exchange a key and then use a faster private key to transfer large amounts of data (e.g. Data Encryption Standards or DES for short which is a block cipher). More sophisticated schemes can also involve a trusted third party which can verify the authenticity of either party (e.g. Verisign).

Having learned some details about cryptography at a level above the average, one might become increasingly aware of the security threats and ultimately refuse to use various advances of modern society like online payments and Internet banking. Paranoia is never a solution! Although minimal threats do exist, the open society we live in is continuously pushing towards better security standards. Another slippery slope is the use of strong cryptography for non-peaceful purposes, and this is what agencies like NSA and GCHQ are trying to control. This is dual position, in the sense that both sides can be argues very extensively and one can hardly ever reach a meaningful conclusion. The best way to address such issues would be to install better awareness of the subject cryptography, so that people cannot be mislead by the numerous sophistic arguments that flow in the public domain.

References

[1] Whitfield Diffie and Martin E. Hellman. New directions in cryptography. *IEEE Transactions on Information Theory*, IT-22(6):644–654, 1976.

[2] J. B. Fraleigh. *A First Course in Abstract Algebra.* Addison-Wesley Publishing Company, Reading, MA, 1982.

[3] D. R. Hofstadter. *Godel, Escher, Bach: an Eternal Golden Braid.* Basic Books, NY, 1979.

[4] T. Jech. *Set Theory.* Springer, NY, 2006.

[5] I. Niven and H. Zuckerman. *An Introduction to the Theory of Numbers.* Wiley, New York, fifth edition, 1991.